University of Málaga

Health Engineering

Assigment

Advanced Models

Author

Alejandro Domínguez Recio

Course

Intelligent Systems

Teachers

Enrique Domínguez Merino

Jesús de Benito Picazo

Introduction

In this assigment we going to evaluate one dataset using Neural Networks, Support Vector Machines and Nearest Neightbor. Use the implementations of these models provided in Weka (MultilayerPerceptron, SMO, IBk) tuning their parameters in order to both improve the performance and obtain the best results. Compare the results of each model, providing a comparative table and/or figure, and justifying the selection of the best model when the entire dataset is used for training.

Data characteristics

- **Dataset context**

The data that we are going to use comes from the analysis of biomechanical characteristics of orthopedic patients. From these we are going to create three models using neural networks, support vector machines and nearest neightbor respectively in order to make a diagnosis from a vector of attributes.

- **Number of classes**

The classes will depend on the type of diagnosis made, of which we have three possible cases: Normal, Disk Hernia or Spondylolisthesis, so we face a multiclass problem.

- **Number of attributes**

We have six biomechanical attributes for each patient plus the class one.

- **Number of samples or instances**

Total instances or samples is 310

➔ **Configuration for neural networks**

Configurations

Configuration	HiddenLayers	LearningRate	Momentum
C1	1	0,3	0,3
C2	1	0,01	0,01
C3	1	0,8	0,8
C4	1	0,8	0,01
C5	2	0,3	0,3
C6	2	0,01	0,01
C7	2	0,8	0,8
C8	2	0,8	0,01

HiddenLayers

The number of layers will depend on the characteristics of the function that we want to represent with our neural network. For cases in which we want to represent linear functions, probably with a single layer it will be enough although in theory with two layers the total of the functions could be represented practically. In our case we are using a multilayer perceptron allowing us to represent more complex functions. When looking for the ideal number of layers we can opt for several strategies such as random or systematic experimentation, orientation by similar networks already implemented or simply intuition by the characteristics of the dataset in question.

LearningRate

Assuming that neural networks are trained using the stochastic descending gradient algorithm. This estimates the gradient error between the current state of the model and the result, which is used to update the model weights using backpropagation of the errors. The learning ratio, a value between 0 and 1 marks the amount of update of that error in the weights. A ratio of 0 marks a null learning from this error and a learning of 1 marks the total. We emphasize that a learning of 1, although it may seem a priori the most optimal, this creates an unbalanced learning by assigning the total learning to the model used in question, ignoring the rest of the options.

Momentum

Taking into account that the networks use the descending gradient algorithm to minimize the error in reaching a global minimum or what is the same, the parameters where our model presents greater accuracy. The use of the moment (value between 0 and 1) allows that in the search for the global minimum the algorithm does not get stuck in local minima of our possible function. Values close to 0 mark that our steps towards the minimum of our function are null and, on the contrary, values close to 1 mark large steps. We emphasize that this value is added to the product of the learning rate * weightgradient.

➔ Configuration for Support Vector Machines

Configurations

Configuration	Exponent	Gamma
PolyKernelC1	1	
PolyKernelC2	2	
PolyKernelC3	20	
NormalizedC1	1	
NormalizedC2	2	
NormalizedC3	20	
RBFC1		0,01
RBFC2		0,1
RBFC3		1

Exponent

The only parameter that we will modify in the polynomial and normalized kernels will be the exponent which marks the degree of the polynomial. In the case of degree 1 we will have a linear kernel, in the case of degree 2 we will have a quadratic kernel and so on.

Gamma

In the cases of RBF kernel the parameter that we will modify will be gamma which controls the amplitude of the kernel.

➔ Configuration for Nearest Neighbor

Configurations

DistanceFunction\K	1	4	8	7
Euclidean	C1	C2	C3	C4
Minkowski	C5	C6	C7	C8
Manhattan	C9	C10	C11	C12

Choice of K

The parameter k influences the stability of the predictions. When we give the value of K = 1 we are saying that we assign the closest class to the point to predict without counting on the rest, so in the case that we have several points in a relative closeness our prediction will not be of quality. In the cases of K close to the number of instances that we have, what happens is that the prediction will be based on the proportions of these without specifying by location. In order to avoid ties, an odd value of k is usually used.

Distance function

The objective of the distance functions is to measure the relative distance between two points in the problem domain. The data can vary in format, such as real, Boolean, categorical or ordinal numbers, therefore the choice of the measurement function will be associated with the type of data we are dealing with. Normalizing attribute values is a common practice to avoid dominance over the result of excessively large values. Specifying about the chosen measures we can say that Euclidean and Manhattan distance are used for both real and integer values with the difference that the first does not work so well with the increase in dimensionality on the contrary of the second, in turn this is more appropriate in cases where the points are distributed as a grid. In the case of Minkowski it is a generalization of the previous two with the addition of a parameter p. Modifying the value of p we have that with p = 1 it is equal to Manhattan, p = 2 Euclidean and with high values it is associated with Chebychev distance.

- ***Performance metric values by configuration***

Accuracy

We can obtain the acurracy in two ways, one of them is directly from the classifier out and the another one is by calculating the number of correctly predicted examples divided by the total number of examples.

Precision

This measurement shows the positive predictive value, higher is bettter. We are going to obtain the necessary values from the confusion matrix and perform the following calculation TP/(TP+FP).

Fallout

This measurement shows the false positive rate, lower is better. We are going to obtain the necessary values from the confusion matrix and perform the following calculation FP/(FP+TN).

Recall

This measurement shows the true positive rate, higher is better. We are going to obtain the necessary values from the confusion matrix and perform the following calculation TP/(TP+FN).

F-measure

This measurement provides a single score that balances both the concers of precision and recall in one number, higher is better. We are going to obtain the necessary values of the previous measurements, precision and recall, and perform the following calculation 2*(Precision*Recall)/(Precision+Recall).

- Results for neural networks

Configurations	Acurracy	Fallout	Precision	Recall	F-measure	ROC	Time
C1	77,096	0,116	0,658	0,771	0,702	0,89	0,3
C2	71,290	0,169	?	0,713	?	0,86	0,08
C3	73,871	0,117	0,721	0,739	0,723	0,89	0,08
C4	77,096	0,111	0,757	0,771	0,738	0,89	0,08
C5	85,1613	0,064	0,853	0,852	0,852	0,96	0,15
C6	70,9677	0,172	?	0,710	?	0,86	0,15
C7	83,2258	0,071	0,835	0,832	0,834	0,93	0,14
C8	84,2258	0,064	0,849	0,845	0,846	0,95	0,12

After obtaining the results, we can conclude that the configurations that give the best performance are those made up of two hiddenlayers, except for C6 which, perhaps due to the small value of both the learning rate and the momentum, loses some performance. I would like to add that the number of hidden layers does not have a relationship directly proportional to performance, although in this case it does happen because if we increase this value too much we can overfit our model. The execution times are also longer in the configurations with two layers, although this was to be expected since the greater the number of layers, the greater the number of operations to be carried out, however, as in the case of the number of hidden layers, we except the cases of overfitting and under fitting. I would also like to highlight the effect produced by the learning rate and momentum, not being excessively noticeable the difference between the configurations with the same number of layers from what we can say, being able to say that said problem does not contain a large number of local minimums in which the gradient can be 'trapped', however, as we said before, configuration 6 shows worse learning, perhaps due to an excessively small value of these, slowing down both its progress and learning.

- Results for Support Vector Machines

Configurations	Acurracy	Fallout	Precision	Recall	F-measure	ROC	Time
C1	76,4516	0,137	0,776	0,765	0,755	0,85	0,06
C2	77,0968	0,121	0,782	0,771	0,768	0,86	0,13
C3	57,7419	0,202	0,662	0,577	0,535	0,69	0,06
C4	77,4194	0,144	0,774	0,774	0,761	0,84	0,1
C5	78,7097	0,123	0,781	0,787	0,780	0,86	0,05
C6	79,3548	0,110	0,789	0,794	0,791	0,86	0,06
C7	48,3871	0,484	?	0,484	?	0,50	0,1
C8	67,0968	0,210	?	0,671	?	0,77	0,05
C9	79,0323	0,111	0,791	0,790	0,789	0,87	0,04

Regarding the results, we can say that Normalized poly kernel is the one that possibly gives us the best result due to the normalization of the values since these can excessively large variations that influence the classification. At the same time, we also note that the higher the exponent in Normalized kernel and the higher the gamma in RBF the results improve, this may be due to the dimension of our characteristics requiring greater differentiation. In poly kernel the latter does not happen possibly due to the increase of the exponent without the normalization of the data thus creating greater distortion between them.

- Results for Nonparametric Models

Configurations	Acurracy	Fallout	Precision	Recall	F-measure	ROC	Time
C1	77,7419	0,117	0,782	0,777	0,778	0,82	0
C2	77,7419	0,117	0,782	0,777	0,778	0,82	0
C3	75,8065	0,124	0,766	0,758	0,761	0,80	0
C4	75,4839	0,120	0,759	0,755	0,757	0,89	0
C5	75,1613	0,130	0,753	0,752	0,749	0,89	0
C6	75,1613	0,130	0,753	0,752	0,749	0,89	0
C7	74,5461	0,120	0,755	0,745	0,747	0,90	0
C8	75,4839	0,120	0,759	0,755	0,757	0,89	0
C9	76,4516	0,112	0,771	0,765	0,767	0,89	0
C10	76,4516	0,112	0,771	0,765	0,767	0,89	0
C11	77,7419	0,106	0,783	0,777	0,779	0,91	0
C12	78,0645	0,106	0,785	0,781	0,783	0,91	0

In this case, as we can see, the results do not vary excessively regardless of the value of k, although the configurations that show better performance are those in which the distance function is Manhattan and specifically with K = 7. Although the difference between the C11 and C12 configurations is minimal, we highlight that the last one has odd k and this could be the reason for its slight improvement.

- **Best results**

Configurations	Acurracy	Fallout	Precision	Recall	F-measure	ROC	Time
NN - C5	85,1613	0,064	0,853	0,852	0,852	0,96	0,15
SPM - C9	79,0323	0,111	0,791	0,790	0,789	0,87	0,04
NPM - C12	78,0645	0,106	0,785	0,781	0,783	0,91	0

- **Last conclusions**

In conclusion, after giving these three topics of advanced models, we have had a first contact with such revolutionary and powerful technologies as neural networks, or with a classifier as widely used as KNN as well as with SVM very suitable for classification for multidimensional models. Going deeper into these we have been able to verify the importance of configuring the number of layers of a neural network, taking into account both the function that we want to approximate or the set of data that we are treating, as well as the advantage that kernels offer us at the same time to transform the spaces and convert a non-linearly separable problem into one that is yes or also the correct choice of k so that the resulting model does not suffer from overfitting or underfitting. In short, a door has been opened for us through which we can delve into and explore the different options that these models represent and the results that we can achieve.